Incorporating Glosses into Neural Word Sense Disambiguation

Fuli Luo, Tianyu Liu, Qiaolin Xia, Baobao Chang and **Zhifang Sui**
Key Laboratory of Computational Linguistics, Ministry of Education,
School of Electronics Engineering and Computer Science, Peking University, Beijing, China
{luofuli, tianyu0421, xql, chbb, szf}@pku.edu.cn

Abstract

Word Sense Disambiguation (WSD) aims to identify the correct meaning of polysemous words in the particular context. Lexical resources like WordNet which are proved to be of great help for WSD in the knowledge-based methods. However, previous neural networks for WSD always rely on massive labeled data (context), ignoring lexical resources like glosses (sense definitions). In this paper, we integrate the context and glosses of the target word into a unified framework in order to make full use of both labeled data and lexical knowledge. Therefore, we propose **GAS**: a **g**loss-**a**ugmented W**S**D neural network which jointly encodes the context and glosses of the target word. GAS models the semantic relationship between the context and the gloss in an improved memory network framework, which breaks the barriers of the previous supervised methods and knowledge-based methods. We further extend the original gloss of word sense via its semantic relations in WordNet to enrich the gloss information. The experimental results show that our model outperforms the state-of-the-art systems on several English all-words WSD datasets.

1 Introduction

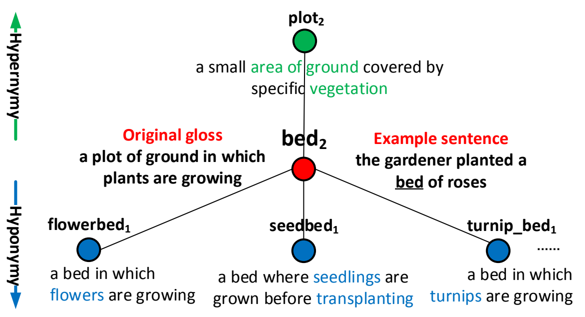
Word Sense Disambiguation (WSD) is a fundamental task and long-standing challenge in Natural Language Processing (NLP). There are several lines of research on WSD. Knowledge-based methods focus on exploiting lexical resources to infer the senses of word in the context. Supervised methods usually train multiple classifiers with manual designed features. Although supervised methods can achieve the state-of-the-art performance (Raganato et al., 2017b,a), there are still two major challenges.

Firstly, supervised methods (Zhi and Ng, 2010; Iacobacci et al., 2016) usually train a dedicated classifier for each word individually (often called *word expert*). So it can not easily scale up to all-words WSD task which requires to disambiguate all the polysemous word in texts [1]. Recent neural-based methods (Kågebäck and Salomonsson, 2016; Raganato et al., 2017a) solve this problem by building a unified model for all the polysemous words, but they still can't beat the best *word expert* system.

Secondly, all the neural-based methods always only consider the local context of the target word, ignoring the lexical resources like WordNet (Miller, 1995) which are widely used in the knowledge-based methods. The gloss, which extensionally defines a word sense meaning, plays a key role in the well-known Lesk algorithm (Lesk, 1986). Recent studies (Banerjee and Pedersen, 2002; Basile et al., 2014) have shown that enriching gloss information through its semantic relations can greatly improve the accuracy of Lesk algorithm.

To this end, our goal is to incorporate the gloss information into a unified neural network for all of the polysemous words. We further consider extending the original gloss through its semantic relations in our framework. As shown in Figure 1, the glosses of hypernyms and hyponyms can enrich the original gloss information as well as help to build better a sense representation. Therefore, we integrate not only the original gloss but also the related glosses of hypernyms and hyponyms into the neural network.

[1] If there are N polysemous words in texts, they need to train N classifiers individually.



Hypernymy
Hyponymy
plot$_2$
a small area of ground covered by specific vegetation
Original gloss
a plot of ground in which plants are growing
bed$_2$
Example sentence
the gardener planted a bed of roses
flowerbed$_1$
seedbed$_1$
turnip_bed$_1$
......
a bed in which flowers are growing
a bed where seedlings are grown before transplanting
a bed in which turnips are growing

Figure 1: The hypernym (green node) and hyponyms (blue nodes) for the 2nd sense bed_2 of bed, which means *a plot of ground in which plants are growing*, rather than the bed for sleeping in. The figure shows that bed_2 is a kind of $plot_2$, and bed_2 includes $flowerbed_1$, $seedbed_1$, etc.

In this paper, we propose a novel model **GAS**: a gloss-augmented WSD neural network which is a variant of the memory network (Sukhbaatar et al., 2015b; Kumar et al., 2016; Xiong et al., 2016). GAS jointly encodes the context and glosses of the target word and models the semantic relationship between the context and glosses in the memory module. In order to measure the inner relationship between glosses and context more accurately, we employ multiple passes operation within the memory as the re-reading process and adopt two memory updating mechanisms.

The main contributions of this paper are listed as follows:

- To the best of our knowledge, our model is the first to incorporate the glosses into an end-to-end neural WSD model. In this way, our model can benefit from not only massive labeled data but also rich lexical knowledge.
- In order to model semantic relationship of context and glosses, we propose a gloss-augmented neural network (GAS) in an improved memory network paradigm.
- We further expand the gloss through its semantic relations to enrich the gloss information and better infer the context. We extend the gloss module in GAS to a hierarchical framework in order to mirror the hierarchies of word senses in WordNet.
- The experimental results on several English all-words WSD benchmark datasets show that our model outperforms the state-of-the-art systems.

2 Related Work

Knowledge-based, supervised and neural-based methods have already been applied to WSD task (Navigli, 2009).

Knowledge-based WSD methods mainly exploit two kinds of knowledge to disambiguate polysemous words: **1)** The gloss, which defines a word sense meaning, is mainly used in Lesk algorithm (Lesk, 1986) and its variants. **2)** The structure of the semantic network, whose nodes are synsets [2] and edges are semantic relations, is mainly used in graph-based algorithms (Agirre et al., 2014; Moro et al., 2014).

Supervised methods (Zhi and Ng, 2010; Iacobacci et al., 2016) usually involve each target word as a separate classification problem (often called *word expert*) and train classifiers based on manual designed features.

Although *word expert* supervised WSD methods perform best in terms of accuray, they are less flexible than knowledge-based methods in the all-words WSD task (Raganato et al., 2017a). To deal with this problem, recent neural-based methods aim to build a unified classifier which shares parameters among all the polysemous words. Kågebäck and Salomonsson (2016) leverages the bidirectional long short-term memory network which shares model parameters among all the polysemous words. Raganato et al. (2017a) transfers the WSD problem into a neural sequence labeling task. However, none of the neural-based methods can totally beat the best *word expert* supervised methods on English all-words WSD datasets.

What's more, all of the previous supervised methods and neural-based methods rarely take the lexical resources like WordNet (Fellbaum, 1998) into consideration. Recent studies on sense embeddings have proved that lexical resources are helpful. Chen et al. (2015) trains word sense embeddings through learning sentence level embeddings from glosses using a convolutional neural networks. Rothe and Schütze (2015) extends word embeddings to sense embeddings by using the constraints and semantic relations in WordNet. They achieve an improvement of more than 1% in WSD performance when using sense embeddings as WSD features for SVM classifier. This work shows that integrating structural information of lexical resources can help to *word expert* supervised methods. However, sense embeddings

[2]A synset is a set of words that denote the same sense.

can only indirectly help to WSD (as SVM classifier features). Raganato et al. (2017a) shows that the coarse-grained semantic labels in WordNet can help to WSD in a multi-task learning framework. As far as we know, there is no study directly integrates glosses or semantic relations of the WordNet into an end-to-end model.

In this paper, we focus on how to integrate glosses into a unified neural WSD system. Memory network (Sukhbaatar et al., 2015b; Kumar et al., 2016; Xiong et al., 2016) is initially proposed to solve question answering problems. Recent researches show that memory network obtains the state-of-the-art results in many NLP tasks such as sentiment classification (Li et al., 2017) and analysis (Gui et al., 2017), poetry generation (Zhang et al., 2017), spoken language understanding (Chen et al., 2016), etc. Inspired by the success of memory network used in many NLP tasks, we introduce it into WSD. We make some adaptations to the initial memory network in order to incorporate glosses and capture the inner relationship between the context and glosses.

3 Incorporating Glosses into Neural Word Sense Disambiguation

In this section, we first give an overview of the proposed model **GAS**: a **g**loss-**a**ugmented W**S**D neural network which integrates the context and the glosses of the target word into a unified framework. After that, each individual module is described in detail.

3.1 Architecture of GAS

The overall architecture of the proposed model is shown in Figure 2. It consists of four modules:

- **Context Module**: The context module encodes the local context (a sequence of surrounding words) of the target word into a distributed vector representation.

- **Gloss Module**: Like the context module, the gloss module encodes all the glosses of the target word into a separate vector representations of the same size. In other words, we can get $|s_t|$ word sense representations according to $|s_t|$ [3] senses of the target word, where $|s_t|$ is the sense number of the target word w_t .

[3] s_t is the sense set $\{s_t^1, s_t^2, \dots, s_t^p\}$ corresponding to the target word x_t

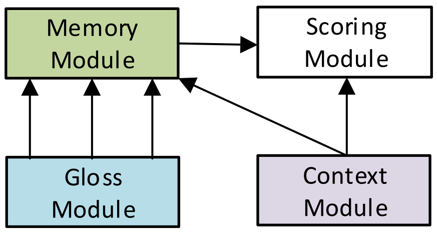


Figure 2: Overview of Gloss-augmented Memory Network for Word Sense Disambiguation.

- **Memory Module**: The memory module is employed to model the semantic relationship between the context embedding and gloss embedding produced by context module and gloss module respectively.

- **Scoring Module**: In order to benefit from both labeled contexts and gloss knowledge, the scoring module takes the context embedding from context module and the last step result from the memory module as input. Finally it generates a probability distribution over all the possible senses of the target word.

Detailed architecture of the proposed model is shown in Figure 3. The next four sections will show detailed configurations in each module.

3.2 Context Module

Context module encodes the context of the target word into a vector representation, which is also called context embedding in this paper.

We leverage the bidirectional long short-term memory network (Bi-LSTM) for taking both the preceding and following words of the target word into consideration. The input of this module $[x_1, \dots, x_{t-1}, x_{t+1}, \dots, x_{T_x}]$ is a sequence of words surrounding the target word x_t, where T_x is the length of the context. After applying a lookup operation over the pre-trained word embedding matrix $\mathrm{M} \in \mathbb{R}^{D \times V}$, we transfer a one hot vector x_i into a D-dimensional vector. Then, the forward LSTM reads the segment $(x_1, \dots, x_{t-1})$ on the left of the target word x_t and calculates a sequence of *forward hidden states* $(\overrightarrow{h}_1, \dots, \overrightarrow{h}_{t-1})$. The backward LSTM reads the segment $(x_{T_x}, \dots, x_{t+1})$ on the right of the target word x_t and calculates a sequence of *backward hidden states* $(\overleftarrow{h}_{T_x}, \dots, \overleftarrow{h}_{t+1})$. The context vector c is finally concatenated as

$$c = [\overrightarrow{h}_{t-1} : \overleftarrow{h}_{t+1}] \tag{1}$$

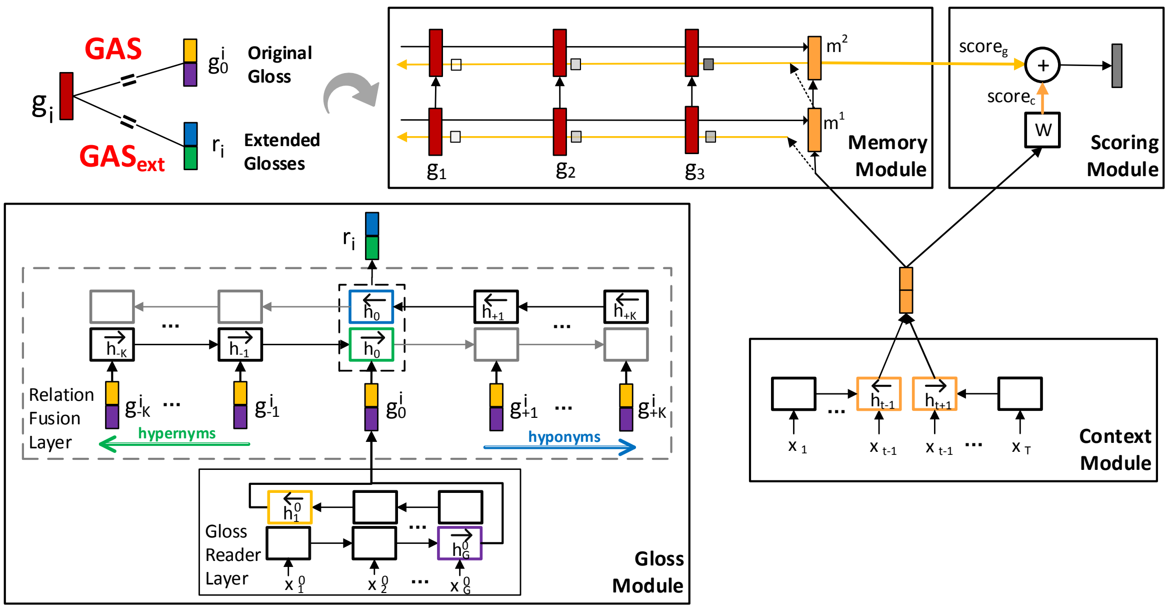


Figure 3: Detailed architecture of our proposed model, which consists of a context module, a gloss module, a memory module and a scoring module. The context module encodes the adjacent words surrounding the target word into a vector c. The gloss module encodes the original gloss or extended glosses into a vector g_i. In the memory module, we calculate the inner relationship (as attention) between context c and each gloss g_i and then update the memory as m_i at pass i. In the scoring module, we make final predictions based on the last pass attention of memory module and the context vector c. Note that GAS only uses the original gloss, while GAS$_{ext}$ uses the entended glosses through hypernymy and hyponymy relations. In other words, the relation fusion layer (grey dotted box) only belongs to GAS$_{ext}$.

where : is the concatenation operator.

3.3 Gloss Module

The gloss module encodes each gloss of the target word into a fixed size vector like the context vector c, which is also called gloss embedding. We further enrich the gloss information by taking semantic relations and their associated glosses into consideration.

This module contains a gloss reader layer and a relation fusion layer. Gloss reader layer generates a vector representations for a gloss. Relation fusion layer aims at modeling the semantic relations of each gloss in the expanded glosses list which consists of related glosses of the original gloss. Our model GAS with extended glosses is denoted as GAS$_{ext}$. GAS only encodes the original gloss, while GAS$_{ext}$ encodes the expanded glosses from hypernymy and hyponymy relations (details in Figure 3).

3.3.1 Gloss Reader Layer

Gloss reader layer contains two parts: gloss expansion and gloss encoder. Gloss expansion is to enrich the original gloss information through its hypernymy and hyponymy relations in WordNet. Gloss encoder is to encode each gloss into a vector representation.

Gloss Expansion: We only expand the glosses of nouns and verbs via their corresponding hypernyms and hyponyms. There are two reasons: One is that most of polysemous words (about 80%) are nouns and verbs; the other is that the most frequent relations among word senses for nouns and verbs are the hypernymy and hyponymy relations [4].

The original gloss is denoted as g_0. Breadth-first search method with a limited depth K is employed to extract the related glosses. The glosses of hypernyms within K depth are denoted as $[g_{-1}, g_{-2}, \ldots, g_{-L_1}]$. The glosses of hyponyms within K depth are denoted as $[g_{+1}, g_{+2}, \ldots, g_{+L_2}]$ [5]. Note that g_{+1} and g_{-1} are the glosses of the nearest word sense.

Gloss Encoder: We denote the j-th [6] gloss in

[4]In WordNet, more than 95% of relations for nouns and 80% for verbs are hypernymy and hyponymy relations.

[5]Since one synset has one or more direct hypernyms and hyponyms, $L_1 >= K$ and $L_2 >= K$.

[6]Since GAS don't have gloss expansion, j is always 0 and $g_i = g_0^i$. See more in Figure 3.

the expanded glosses list for i_{th} sense of the target word as a sequence of G words. Like the context encoder, the gloss encoder also leverages Bi-LSTM units to process the words sequence of the gloss. The gloss representation g_j^i is computed as the concatenation of the last hidden states of the *forward* and *backward* LSTM.

$$g_j^i = [\overrightarrow{h}_G^{i,j} : \overleftarrow{h}_1^{i,j}] \tag{2}$$

where $j \in [-L_1, \ldots, -1, 0, +1, \ldots, +L_2]$ and : is the concatenation operator .

3.3.2 Relation Fusion Layer

Relation fusion layer models the hypernymy and hyponymy relations of the target word sense. A forward LSTM is employed to encode the hypernyms' glosses of i_{th} sense $(g_{-L_1}^i, \cdots, g_{-1}^i, g_0^i)$ as a sequence of *forward hidden states* $(\overrightarrow{h}_{-L_1}^i, \ldots, \overrightarrow{h}_{-1}^i, \overrightarrow{h}_0^i)$. A backward LSTM is employed to encode the hyponyms' glosses of i_{th} sense $(g_{+L_2}^i, \ldots, g_{+1}^i, g_0^i)$ as a sequence of *backward hidden states* $(\overleftarrow{h}_{+L_2}^i, \ldots, \overleftarrow{h}_{+1}^i, \overleftarrow{h}_0^i)$. In order to highlight the original gloss g_0^i, the enhanced i_{th} sense representation is concatenated as the final state of the forward and backward LSTM.

$$g_i = [\overrightarrow{h}_0^i : \overleftarrow{h}_0^i] \tag{3}$$

3.4 Memory Module

The memory module has two inputs: the context vector c from the context module and the gloss vectors $\{g_1, g_2, \ldots, g_{|s_t|}\}$ from the gloss module, where $|s_t|$ is the number of word senses. We model the inner relationship between the context and glosses by attention calculation. Since one-pass attention calculation may not fully reflect the relationship between the context and glosses (details in Section 4.4.2), the memory module adopts a repeated deliberation process. The process repeats reading gloss vectors in the following passes, in order to highlight the correct word sense for the following scoring module by a more accurate attention calculation. After each pass, we update the memory to refine the states of the current pass. Therefore, memory module contains two phases: attention calculation and memory update.

Attention Calculation: For each pass k, the attention e_i^k of gloss g_i is generally computed as

$$e_i^k = f(g_i, m^{k-1}, c) \tag{4}$$

where m^{k-1} is the memory vector in the $(k-1)$-th pass while c is the context vector. The scoring function f calculates the semantic relationship of the gloss and context, taking the vector set (g_i, m^{k-1}, c) as input. In the first pass, the attention reflects the similarity of context and each gloss. In the next pass, the attention reflects the similarity of adapted memory and each gloss. A dot product is applied to calculate the similarity of each gloss vector and context (or memory) vector. We treat c as m^0. So, the attention α_i^k of gloss g_i at pass k is computed as a dot product of g_i and m^{k-1}:

$$e_i^k = g_i \cdot m^{k-1} \tag{5}$$

$$\alpha_i^k = \frac{\exp(e_i^k)}{\sum_{j=1}^{|s_t|} \exp(e_i^j)} \tag{6}$$

Memory Update: After calculating the attention, we store the memory state in u^k which is a weighted sum of gloss vectors and is computed as

$$u^k = \sum_{i=1}^{n} \alpha_i^k g_i \tag{7}$$

where n is the hidden size of LSTM in the context module and gloss module. And then, we update the memory vector m^k from last pass memory m^{k-1}, context vector c, and memory state u^k. We propose two memory update methods:

- *Linear*: we update the memory vector m^k by a linear transformation from m^{k-1}

  $$m^k = Hm^{k-1} + u^k \tag{8}$$

  where $H \in \mathbb{R}^{2n \times 2n}$.

- *Concatenation*: we get a new memory for k-th pass by taking both the gloss embedding and context embedding into consideration

  $$m^k = ReLU(W[m^{k-1} : u^k : c] + b) \tag{9}$$

  where : is the concatenation operator, $W \in \mathbb{R}^{n \times 6n}$ and $b \in \mathbb{R}^{2n}$.

3.5 Scoring Module

The scoring module calculates the scores for all the related senses $\{s_t^1, s_t^2, \ldots, s_t^p\}$ corresponding to the target word x_t and finally outputs a sense probability distribution over all senses.

The overall score for each word sense is determined by gloss attention $\alpha_i^{T_M}$ from the last pass in the memory module, where T_M is the number of passes in the memory module. The e^{T_M} (α^{T_M} without Softmax) is regarded as the gloss score.

$$score_g = e^{T_M} \tag{10}$$

Meanwhile, a fully-connected layer is employed to calculate the context score.

$$score_c = W_{x_t} c + b_{x_t} \tag{11}$$

where $W_{x_t} \in \mathbb{R}^{|s_t| \times 2n}$, $b_{x_t} \in \mathbb{R}^{|s_t|}$, $|s_t|$ is the number of senses for the target word x_t and n is the number of hidden units in the LSTM.

It's noteworthy that in Equation 11, each ambiguous word x_t has its corresponding weight matrix W_{x_t} and bias b_{x_t} in the scoring module.

In order to balance the importance of background knowledge and labeled data, we introduce a parameter $\lambda \in \mathbb{R}^N$ [7] in the scoring module which is jointly learned during the training process. The probability distribution $\hat{y}$ over all the word senses of the target word is calculated as:

$$\hat{y} = Softmax(\lambda_{x_t} score_c + (1 - \lambda_{x_t}) score_g)$$

where λ_{x_t} is the parameter for word x_t, and $\lambda_{x_t} \in [0, 1]$.

During training, all model parameters are jointly learned by minimizing a standard cross-entropy loss between $\hat{y}$ and the true label y.

4 Experiments and Evaluation

4.1 Dataset

Evaluation Dataset: we evaluate our model on several English all-words WSD datasets. For fair comparison, we use the benchmark datasets proposed by Raganato et al. (2017b) which includes five standard all-words fine-grained WSD datasets from the Senseval and SemEval competitions. They are Senseval-2 (**SE2**), Senseval-3 task 1 (**SE3**), SemEval-07 task 17 (**SE7**), SemEval-13 task 12 (**SE13**), and SemEval-15 task 13 (**SE15**). Following by Raganato et al. (2017a), we choose SE7, the smallest test set as the development (validation) set, which consists of 455 labeled instances. The last four test sets consist of 6798 labeled instances with four types of target words, namely nouns, verbs, adverbs and adjectives. We extract word sense glosses from WordNet3.0 because Raganato et al. (2017b) maps all the sense annotations [8] from its original version to 3.0.

Training Dataset: We choose SemCor 3.0 as the training set, which was also used by Raganato et al. (2017a), Raganato et al. (2017b), Iacobacci et al. (2016), Zhi and Ng (2010), etc. It consists of 226,036 sense annotations from 352 documents, which is the largest manually annotated corpus for WSD. Note that all the systems listed in Table 1 are trained on SemCor 3.0.

4.2 Implementation Details

We use the validation set (SE7) to find the optimal settings of our framework: the hidden state size n, the number of passes $|T_M|$, the optimizer, etc. We use pre-trained word embeddings with 300 dimensions[9], and keep them fixed during the training process. We employ 256 hidden units in both the gloss module and the context module, which means n=256. Orthogonal initialization is used for weights in LSTM and random uniform initialization with range [-0.1, 0.1] is used for others. We assign gloss expansion depth K the value of 4. We also experiment with the number of passes $|T_M|$ from 1 to 5 in our framework, finding $|T_M| = 3$ performs best. We use Adam optimizer (Kingma and Ba, 2014) in the training process with 0.001 initial learning rate. In order to avoid overfitting, we use dropout regularization and set drop rate to 0.5. Training runs for up to 100 epochs with early stopping if the validation loss doesn't improve within the last 10 epochs.

4.3 Systems to be Compared

In this section, we describe several knowledge-based methods, supervised methods and neural-based methods which perform well on the English all-words WSD datasets for comparison.

4.3.1 Knowledge-based Systems

- **Lesk**$_{ext+emb}$: Basile et al. (2014) is a variant of Lesk algorithm (Lesk, 1986) by using a word similarity function defined on a distributional semantic space to calculate the gloss-context overlap. This work shows that glosses are important to WSD and enriching

[7] N is the number of polysemous words in the training corpora.

[8] The original WordNet version of SE2, SE3, SE7, SE13, SE15 are 1.7, 1.7.1, 2.1, 3.0 and 3.0, respectively.

[9] We download the pre-trained word embeddings from `https://github.com/stanfordnlp/GloVe`, and we select the smaller Wikipedia 2014 + Gigaword 5.

	Test Datasets				Concatenation of Test Datasets				
System	SE2	SE3	SE13	SE15	Noun	Verb	Adj	Adv	**All**
MFS baseline	65.6	66.0	63.8	67.1	67.7	49.8	73.1	80.5	65.5
Lesk$_{ext+emb}$ (Basile et al., 2014)	63.0	63.7	66.2	64.6	70.0	51.1	51.7	80.6	64.2
Babelfy (Moro et al., 2014)	67.0	63.5	66.4	70.3	68.9	50.7	73.2	79.8	66.4
IMS (Zhi and Ng, 2010)	70.9	69.3	65.3	69.5	70.5	55.8	75.6	82.9	68.9
IMS$_{+emb}$ (Iacobacci et al., 2016)	72.2	70.4	65.9	71.5	71.9	56.6	75.9	84.7	70.1
Bi-LSTM (Kågebäck and Salomonsson, 2016)	71.1	68.4	64.8	68.3	69.5	55.9	76.2	82.4	68.4
Bi-LSTM$_{+att.+LEX}$ (Raganato et al., 2017a)*	72.0	69.4	66.4	72.4	71.6	57.1	75.6	83.2	69.9
Bi-LSTM$_{+att.+LEX+POS}$ (Raganato et al., 2017a)*	72.0	69.1	66.9	71.5	71.5	57.5	75.0	83.8	69.9
GAS (Linear)*	72.0	70.0	66.7	71.6	71.7	57.4	76.5	83.5	70.1
GAS (Concatenation)*	72.1	70.2	67.0	71.8	72.1	57.2	76.0	84.4	70.3
GAS$_{ext}$ (Linear)*	**72.4**	70.1	67.1	72.1	71.9	**58.1**	76.4	84.7	70.4
GAS$_{ext}$ (Concatenation)*	72.2	**70.5**	**67.2**	**72.6**	**72.2**	57.7	**76.6**	**85.0**	**70.6**

Table 1: F1-score (%) for fine-grained English all-words WSD on the test sets. **Bold** font indicates best systems. The * represents the neural network models using external knowledge. The fives blocks list the MFS baseline, two knowledge-based systems, two supervised systems (feature-based), three neural-based systems and our models, respectively.

gloss information via its semantic relations can help to WSD.

- **Babelfy**: Moro et al. (2014) exploits the semantic network structure from BabelNet and builds a unified graph-based architecture for WSD and Entity Linking.

4.3.2 Supervised Systems

The supervised systems mentioned in this paper refers to traditional feature-based systems which train a dedicated classifier for every word individually (*word expert*).

- **IMS**: Zhi and Ng (2010) selects a linear Support Vector Machine (SVM) as its classifier and makes use of a set of features surrounding the target word within a limited window, such as POS tags, local words and local collocations.

- **IMS$_{+emb}$**: Iacobacci et al. (2016) selects IMS as the underlying framework and makes use of word embeddings as features which makes it hard to beat in most of WSD datasets.

4.3.3 Neural-based Systems

Neural-based systems aim to build an end-to-end unified neural network for all the polysemous words in texts.

- **Bi-LSTM**: Kågebäck and Salomonsson (2016) leverages a bidirectional LSTM network which shares model parameters among all words. Note that this model is equivalent to our model if we remove the gloss module and memory module of GAS.

- **Bi-LSTM$_{+att.+LEX}$** and its variant **Bi-LSTM$_{+att.+LEX+POS}$**: Raganato et al. (2017a) transfers WSD into a sequence learning task and propose a multi-task learning framework for WSD, POS tagging and coarse-grained semantic labels (LEX). These two models have used the external knowledge, for the LEX is based on lexicographer files in WordNet.

Moreover, we introduce **MFS** baseline, which simply selects the most frequent sense in the training data set.

4.4 Results and Discussion

4.4.1 English all-words results

In this section, we show the performance of our proposed model in the English all-words task. Table 1 shows the F1-score results on the four test sets mentioned in Section 4.1. The systems in the first four blocks are implemented by Raganato et al. (2017a,b) except for the single Bi-LSTM model. The last block lists the performance of our proposed model GAS and its variant GAS$_{ext}$ which extends the gloss module in GAS.

GAS and GAS$_{ext}$ achieves the state-of-the-art performance on the concatenation of all test datasets. Although there is no one system always performs best on all the test sets [10], we can find that GAS$_{ext}$ with *concatenation* memory updating strategy achieves the best results **70.6** on the concatenation of the four test datasets. Compared with other three neural-based methods in the

[10] Because the source of the four datasets are extremely different which belongs to different domains.

Context: He **plays** a pianist in the film					
Glosses	Pass 1	Pass 2	Pass 3	Pass 4	Pass 5
g_1: participate in games or sport	░				
g_2: perform music on a instrument	▒	▒	░	░	░
g_3: act a role or part	▒	▓	█	█	█

Table 2: An example of attention weights in the memory module within 5 passes. Darker colors mean that the attention weight is higher. Case studies show that the proposed multi-pass operation can recognize the correct sense by enlarging the attention gap between correct senses and incorrect ones.

Pass	**SE2**	**SE3**	**SE13**	**SE15**	**ALL**
1	71.6	70.3	67.0	72.5	70.3
2	71.9	70.2	67.1	**72.8**	70.4
3	**72.2**	**70.5**	**67.2**	72.6	**70.6**
4	72.1	70.4	**67.2**	72.4	70.5
5	72.0	70.4	67.1	71.5	70.3

Table 3: F1-score (%) of different passes from 1 to 5 on the test data sets. It shows that appropriate number of passes can boost the performance as well as avoid over-fitting of the model.

fourth block, we can find that our best model outperforms the previous best neural network models (Raganato et al., 2017a) on every individual test set. The IMS$_{+emb}$, which trains a dedicated classifier for each word individually (*word expert*) with massive manual designed features including word embeddings, is hard to beat for neural networks models. However, our best model can also beat IMS$_{+emb}$ on the SE3, SE13 and SE15 test sets.

Incorporating glosses into neural WSD can greatly improve the performance and extending the original gloss can further boost the results. Compared with the Bi-LSTM baseline which only uses labeled data, our proposed model greatly improves the WSD task by **2.2%** F1-score with the help of gloss knowledge. Furthermore, compared with the GAS which only uses original gloss as the background knowledge, GAS$_{ext}$ can further improve the performance with the help of the extended glosses through the semantic relations. This proves that incorporating extended glosses through its hypernyms and hyponyms into the neural network models can boost the performance for WSD.

4.4.2 Multiple Passes Analysis

To better illustrate the influence of multiple passes, we give an example in Table 2. Consider the situation that we meet an unknown word **x** [11], we look up from the dictionary and find three word senses and their glosses corresponding to **x**.

[11]**x** refers to word *play* in reality.

We try to figure out the correct meaning of **x** according to its context and glosses of different word senses by the proposed memory module. In the first pass, the first sense is excluded, for there are no relevance between the context and g_1. But the g_2 and g_3 may need repeated deliberation, for word *pianist* is similar to the word *music* and *role* in the two glosses. By re-reading the context and gloss information of the target word in the following passes, the correct word sense g_3 attracts much more attention than the other two senses. Such re-reading process can be realized by multi-pass operation in the memory module.

Furthermore, Table 3 shows the effectiveness of multi-pass operation in the memory module. It shows that multiple passes operation performs better than one pass, though the improvement is not significant. The reason of this phenomenon is that for most target words, one main word sense accounts for the majority of their appearances. Therefore, in most circumstances, one-pass inference can lead to the correct word senses. Case studies in Table 2 show that the proposed multi-pass inference can help to recognize the infrequent senses like the third sense for word *play*. In Table 3, with the increasing number of passes, the F1-score increases. However, when the number of passes is larger than 3, the F1-score stops increasing or even decreases due to over-fitting. It shows that appropriate number of passes can boost the performance as well as avoid over-fitting of the model.

5 Conclusions and Future Work

In this paper, we seek to address the problem of integrating the glosses knowledge of the ambiguous word into a neural network for WSD. We further extend the gloss information through its semantic relations in WordNet to better infer the context. In this way, we not only make use of labeled context data but also exploit the background knowledge to disambiguate the word sense. Results on four English all-words WSD data sets show that our best model outperforms the existing methods.

There is still one challenge left for the future. We just extract the gloss, missing the structural properties or graph information of lexical resources. In the next step, we will consider integrating the rich structural information into the neural network for Word Sense Disambiguation.

Acknowledgments

We thank the Lei Sha, Jiwei Tan, Jianmin Zhang and Junbing Liu for their instructive suggestions and invaluable help. The research work is supported by the National Science Foundation of China under Grant No. 61772040 and No. 61751201. The contact authors are Baobao Chang and Zhifang Sui.

References

Eneko Agirre, Oier Lpez De Lacalle, and Aitor Soroa. 2014. Random walks for knowledge-based word sense disambiguation. *Computational Linguistics* 40(1):57–84.

Satanjeev Banerjee and Ted Pedersen. 2002. An adapted lesk algorithm for word sense disambiguation using wordnet. In *International Conference on Intelligent Text Processing and Computational Linguistics*. Springer, pages 136–145.

Satanjeev Banerjee and Ted Pedersen. 2003. Extended gloss overlaps as a measure of semantic relatedness. In *Ijcai*. volume 3, pages 805–810.

Pierpaolo Basile, Annalina Caputo, and Giovanni Semeraro. 2014. An enhanced lesk word sense disambiguation algorithm through a distributional semantic model. In *Roceedings of COLING 2014, the International Conference on Computational Linguistics: Technical Papers*.

José Camacho-Collados, Mohammad Taher Pilehvar, and Roberto Navigli. 2015. A unified multilingual semantic representation of concepts. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. volume 1, pages 741–751.

T. Chen, R. Xu, Y. He, and X. Wang. 2015. Improving distributed representation of word sense via wordnet gloss composition and context clustering. *Atmospheric Measurement Techniques* 4(3):5211–5251.

Xinxiong Chen, Zhiyuan Liu, and Maosong Sun. 2014. A unified model for word sense representation and disambiguation. In *Conference on Empirical Methods in Natural Language Processing*. pages 1025–1035.

Yun Nung Chen, Dilek Hakkani-Tr, Gokhan Tur, Jianfeng Gao, and Li Deng. 2016. End-to-end memory networks with knowledge carryover for multi-turn spoken language understanding. In *The Meeting of the International Speech Communication Association*.

Christiane Fellbaum. 1998. *WordNet*. Wiley Online Library.

Lin Gui, Jiannan Hu, Yulan He, Ruifeng Xu, Qin Lu, and Jiachen Du. 2017. A question answering approach to emotion cause extraction. *arXiv preprint arXiv:1708.05482* .

Ignacio Iacobacci, Mohammad Taher Pilehvar, and Roberto Navigli. 2016. Embeddings for word sense disambiguation: An evaluation study. In *The Meeting of the Association for Computational Linguistics*.

Mikael Kågebäck and Hans Salomonsson. 2016. Word sense disambiguation using a bidirectional lstm. *arXiv preprint arXiv:1606.03568* .

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *Computer Science* .

Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus, and Richard Socher. 2016. Ask me anything: Dynamic memory networks for natural language processing. In *International Conference on Machine Learning*. pages 1378–1387.

Michael Lesk. 1986. Automatic sense disambiguation using machine readable dictionaries:how to tell a pine cone from an ice cream cone. In *Acm Special Interest Group for Design of Communication*. pages 24–26.

Qi Li, Tianshi Li, and Baobao Chang. 2016. Learning word sense embeddings from word sense definitions .

Zheng Li, Yu Zhang, Ying Wei, Yuxiang Wu, and Qiang Yang. 2017. End-to-end adversarial memory network for cross-domain sentiment classification. In *Twenty-Sixth International Joint Conference on Artificial Intelligence*. pages 2237–2243.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM* 38(11):39–41.

Andrea Moro, Alessandro Raganato, and Roberto Navigli. 2014. Entity linking meets word sense disambiguation: a unified approach. *Transactions of the Association for Computational Linguistics* 2:231–244.

Roberto Navigli. 2009. Word sense disambiguation:a survey. *Acm Computing Surveys* 41(2):1–69.

Alessandro Raganato, Claudio Delli Bovi, and Roberto Navigli. 2017a. Neural sequence learning models for word sense disambiguation. In *Conference on Empirical Methods in Natural Language Processing*.

Alessandro Raganato, Jose Camacho-Collados, and Roberto Navigli. 2017b. Word sense disambiguation: A unified evaluation framework and empirical comparison. In *Proc. of EACL*. pages 99–110.

Sascha Rothe and Hinrich Schütze. 2015. Autoextend: Extending word embeddings to embeddings for synsets and lexemes. *arXiv preprint arXiv:1507.01127* .

Lei Sha, Feng Qian, and Zhifang Sui. 2017. Will repeated reading benefit natural language understanding? In *National CCF Conference on Natural Language Processing and Chinese Computing*. pages 366–379.

Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2015a. End-to-end memory networks. *Computer Science* .

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015b. End-to-end memory networks. In *Advances in neural information processing systems*. pages 2440–2448.

Yingce Xia, Fei Tian, Lijun Wu, Jianxin Lin, Tao Qin, Nenghai Yu, and Tie Yan Liu. 2017. Deliberation networks: Sequence generation beyond one-pass decoding .

Caiming Xiong, Stephen Merity, and Richard Socher. 2016. Dynamic memory networks for visual and textual question answering. In *International Conference on Machine Learning*. pages 2397–2406.

Jiyuan Zhang, Yang Feng, Dong Wang, Yang Wang, Andrew Abel, Shiyue Zhang, and Andi Zhang. 2017. Flexible and creative chinese poetry generation using neural memory pages 1364–1373.

Zhong Zhi and Hwee Tou Ng. 2010. It makes sense: A wide-coverage word sense disambiguation system for free text. In *ACL 2010, Proceedings of the Meeting of the Association for Computational Linguistics, July 11-16, 2010, Uppsala, Sweden, System Demonstrations*. pages 78–83.